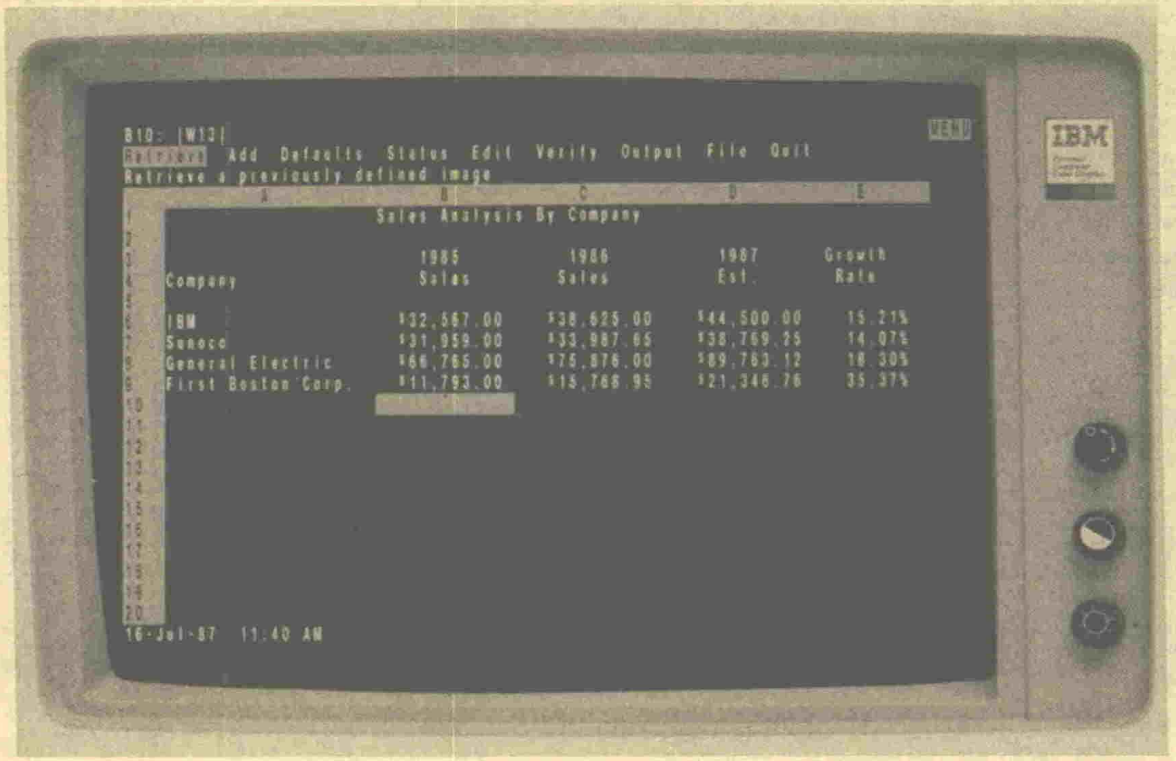
# โครงสร้างข้อมูลพื้นฐานของโปรแกรม สเปรดชีต




**จเร เลิศสุดวิชัย**

เคยสงสัยกันบ้างไหมว่าโปรแกรมสำเร็จรูปประเภทสเปรดชีตที่เราใช้งานกันอยู่ประจำนั้นมีหลักการในการเก็บข้อมูลตั้งมากมายที่เราป้อนเข้าไปอย่างไร จึงได้มีความสามารถทำอะไรต่อมิอะไรได้ตั้งมากมายขนาดนั้นแถมยังรวดเร็วดีเสียด้วย ถ้าเคยสงสัยละก็เนื้อหาในบทความนี้จะเสนอเรื่องราวแนวความคิดทางด้านโครงสร้างข้อมูลพื้นฐานของโปรแกรมสเปรดชีต และวิธีการคำนวณสูตรในสเปรดชีต เนื้อเรื่องในบทความนี้จะต้องอาศัยความรู้ทางด้านโครงสร้างข้อมูล (Data Structure) ด้วย บทความนี้คงช่วยให้ท่านผู้อ่านเข้าใจถึงการทำงานของโปรแกรมสำเร็จรูปสเปรดชีตที่มีให้เลือกใช้หลายโปรแกรมในท้องตลาดขณะนี้

## โครงสร้างข้อมูลของสเปรดชีต

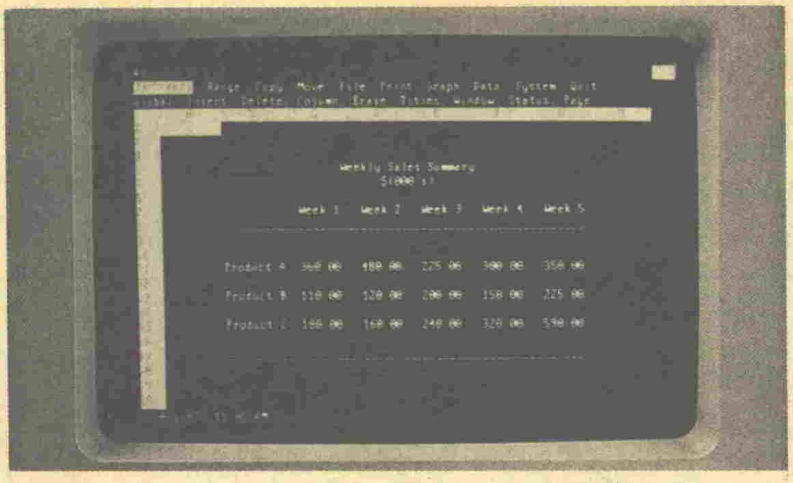
ลักษณะของโปรแกรมสเปรดชีตเป็นตาราง ในแต่ละช่องเรียกว่าเซลล์ (cell) ซึ่งในตัวเซลล์แต่ละเซลล์จะต้องมีโครงสร้างข้อมูลสำหรับเก็บค่าตัวเลขในการคำนวณหรือสูตร หรือข้อความ และอาจจะมีส่วนอื่น ๆ อีกที่จะต้องใช้ช่วยการคำนวณ จากลักษณะของโครงสร้างนี้จะมีโครงสร้างอะไรที่จะเหมาะสมสำหรับการประกอบเซลล์ทุกเซลล์เข้าด้วยกันเป็นสเปรดชีต โดยที่มีการเข้าถึงเซลล์แต่ละเซลล์ได้อย่างรวดเร็ว และใช้หน่วยความจำแบบมีประสิทธิภาพมากที่สุด หลายท่านคงคิดถึงโครงสร้างข้อมูลแบบง่ายคือ โครงสร้างข้อมูลแบบอะเรย์ (array) ของเซลล์ในลักษณะ 2 มิติ (two dimension) ลองคำนวณกันแบบคร่าว ๆ ดู สมมติว่าแต่ละเซลล์ใช้ 1 ไบต์ เพื่อความง่าย (สเปรดชีตของโลตัส 1-2-3 มีจำนวนคอลัมน์ 256 คอลัมน์ × จำนวนแถว 8,192 ส่วนของไมโครซอฟต์เอกเซล (Microsoft Excel) มี 256 คอลัมน์×16384) สมมติว่าเราแทนโครงสร้างข้อมูลของสเปรดชีตด้วยอะเรย์ขนาด 256 คอลัมน์ 1000 แถว จะต้องใช้หน่วยความจำขนาด 256×1000 ไบต์ หรือประมาณ 250 กิโลไบต์ จะเห็นว่าเพียงแค่ขนาดข้อมูลก็กินเนื้อที่หน่วยความจำเกือบหมดแล้ว ถ้าหน่วยความจำบนเครื่องมีขนาด 256 กิโลไบต์ ดังนั้นลักษณะโครงสร้างข้อมูลแบบ static data จึงไม่ถูกนำมาใช้ และจากการใช้งานสเปรด-

ชีตตามปกตินั้นผู้ใช้งานอีกมากที่ไม่ได้ใช้งานเซลล์ทุก ๆ เซลล์ในสเปรดชีต แต่จะใช้ในลักษณะเป็นกลุ่มเซลล์ ดังนั้นจึงไม่จำเป็นต้องจองหน่วยความจำมาเป็นเซลล์ทั้งหมด ซึ่งปัญหานี้การใช้หน่วยความจำแบบ dynamic allocation จึงถูกนำมาใช้งาน ซึ่งลักษณะโครงสร้างข้อมูลของสเปรดชีตที่จะเสนอมีด้วยกันหลายลักษณะด้วยกัน แต่ละลักษณะก็จะมีข้อดีและข้อเสียต่างกัน เช่นลิงค์ลิสต์ (linked list); ไบนารี่ทรี (binary tree) และอะเรย์ของพอยน์เตอร์ (pointer array) ซึ่งในตอนสุดท้ายจะเป็นโครงสร้างข้อมูลที่รวมเอาข้อดีของทั้งหมดมารวมกัน

## โครงสร้างข้อมูลแบบ Doubly linked list

จากปัญหาเรื่องหน่วยความจำ และลักษณะการใช้เซลล์ของสเปรดชีตในลักษณะเป็นกลุ่มเซลล์ไม่ได้ใช้ทั้งหมด ตัวเซลล์แต่ละเซลล์ที่ถูกใช้จะถูกจองที่ (allocate) จากฮีพ (Heap) แล้วเก็บไว้ในลิสต์ (list) โดยจะถูกเรียงตามลำดับชื่อเซลล์ เพื่อสะดวกในการค้นหาหรือเข้าถึงเซลล์ดังรูปที่ 1

จากลักษณะนี้การค้นหาเซลล์จะต้องเดินเข้าไปใน list เป็นการค้นแบบเชิงเส้น (linear search) ซึ่งการค้นหาแบบเชิงเส้นนี้จะต้องทำการเปรียบเทียบชื่อเซลล์ โดยเฉลี่ยของการค้นหาแบบเชิงเส้นนี้ จะมีการเปรียบเทียบ n/2 ครั้ง เมื่อ n คือจำนวน element หรือเซลล์ในลิสต์ โครงสร้างข้อมูลแบบ linked list สามารถจะแก้ปัญหาเรื่องการใช้หน่วยความจำได้ แต่การค้นหาเซลล์หรือเข้าถึงเซลล์จะช้ามาก



## โครงสร้างข้อมูลแบบ Binary Tree

โครงสร้างข้อมูลแบบ Binary Tree นั้นสามารถแก้ปัญหาเรื่องการค้นหาเซลล์หรือการเข้าถึงเซลล์ได้รวดเร็วกว่าโครงสร้างแบบ linked list

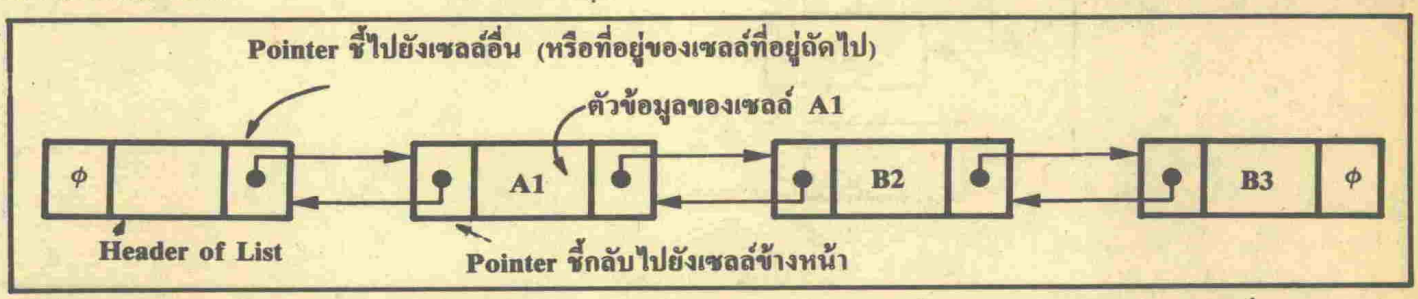
ลักษณะการเรียงต่อของเซลล์ในโครงสร้างข้อมูลแบบ Binary tree นั้น เซลล์ที่จะต่อจาก node ใด ถ้าชื่อเซลล์มีค่าน้อยกว่าจะไปต่อทางซ้าย ถ้ามากกว่าจะต่อทางขวาของ node นั้น ดังในรูปที่ 2 ในการค้นหาเซลล์ในโครงสร้างข้อมูลนี้ โดยเฉลี่ยจะต้องเปรียบเทียบ $\log_2 n$ ครั้ง เมื่อ n คือจำนวนเซลล์ใน tree (ลองเปรียบเทียบกับ linked tist ซึ่งใช้การเปรียบเฉลี่ย n/2 ครั้ง) ซึ่งจะค้นได้รวดเร็วกว่า linked list อย่างไรก็ตามลักษณะโครงสร้างแบบ Binary tree ค่อนข้างซับซ้อนเมื่อมีลบเซลล์หรือเพิ่มเซลล์เข้าไปใน tree จะต้องทำการสมดุลย์ (Balance tree) ในการทำงานบางอย่างของสเปรดชีต เช่นการ copy ข้อมูลเป็น range จะทำได้ค่อนข้างยุ่งยาก

## โครงสร้างข้อมูลแบบ Pointer-arrays

ในโครงสร้างข้อมูลแบบนี้ สิ่งที่อยู่ในอะเรย์ จะเป็นพอยน์เตอร์หรือที่อยู่ของเซลล์ที่เกิดขึ้น เป็นลักษณะอะเรย์ 2 ทิศทาง โดยที่มีตัวชี้เป็นตัวบอกแถวและคอลัมน์ ดังในรูปที่ 3

เมื่อมีการใช้เซลล์ ตัวเซลล์จะถูกจองที่จากฮีพ ที่อยู่ของเซลล์จะถูกเก็บไว้ในอะเรย์ ตัวอย่างเช่น เซลล์ A1 ถูกใช้งาน ที่อยู่ (address) ของเซลล์ A1 ที่ได้จากการจองที่จะถูกนำไปใส่ในอะเรย์ข้อมูล แถวที่ 1 และคอลัมน์ที่ 1 (ดังรูปที่ 3)

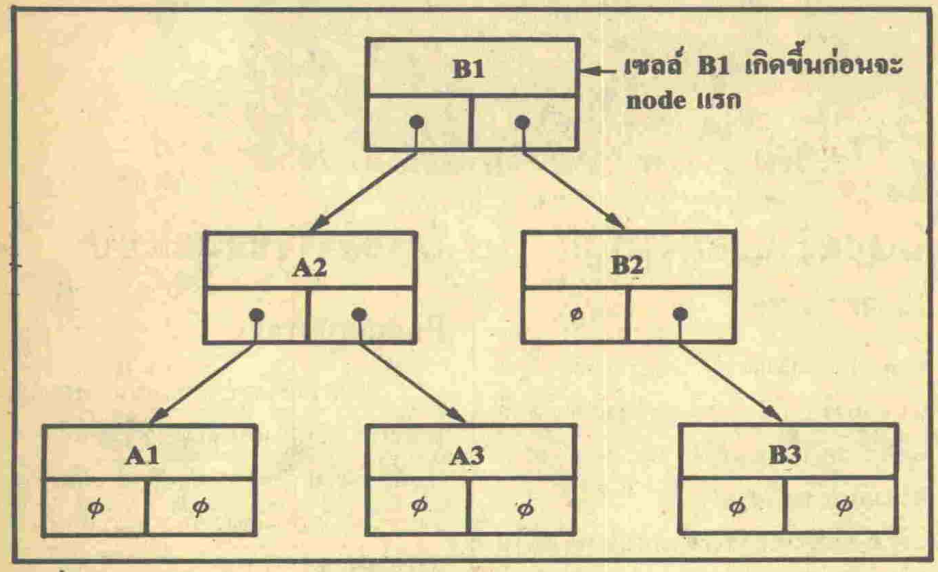
จากลักษณะโครงสร้างอันนี้ การค้นหาเซลล์ทำได้ทันทีเพราะเป็นข้อมูลในลักษณะของอะเรย์ เพียงแต่บอกตัวชี้แถวและคอลัมน์เท่านั้นก็สามารถค้นพบเซลล์ได้ แต่ข้อเสียโครงสร้างอันนี้ก็คือ เนื่องจากมันเป็นโครงสร้างข้อมูลลักษณะอะเรย์ ดังนั้นมันจึงต้องการใช้หน่วยความจำจำนวนหนึ่งเป็นอะเรย์ของพอยน์เตอร์ ดังนั้นการ




รูปที่ 1 โครงสร้างข้อมูลแบบ Linked list แต่ละ element ใน list แทนข้อมูลเซลล์ในสเปรดชีต โดยเรียงลำดับชื่อของเซลล์

ใช้โครงสร้างข้อมูลชนิดนี้จึงจำเป็นต้องกำหนดหน่วยความจำที่ใช้เป็นอะเรย์ที่มีขนาดที่แน่นอน โดยเหลือหน่วยความจำจำนวนหนึ่งไว้เป็นส่วนของตัวเซลล์ นั่นคือขนาดของสเปรดชีตจะถูกจำกัดอยู่ที่จำนวนแถวและคอลัมน์ที่แน่นอน เราลองคำนวณคร่าว ๆ ดู สมมติเราต้องการสร้างสเปรดชีตขนาด 100 แถว และ 100 คอลัมน์จะต้องใช้เนื้อที่ถูกจองเป็นเนื้อที่ถูกจองเป็นอะเรย์เท่าใด โดยที่พอยน์เตอร์หรือที่อยู่ของเซลล์ประกอบด้วย เซกเมนต์และออฟเซต (segment:offset, การอ้างแอดเดรสในซีพียูเบอร์ 8088-8086) รวมเป็น 4 ไบต์

*ดังนั้นจำนวนหน่วยความจำที่ต้องใช้*
*= 100×100×4*
*= 40,000 ไบต์ หรือ 39 กิโลไบต์*



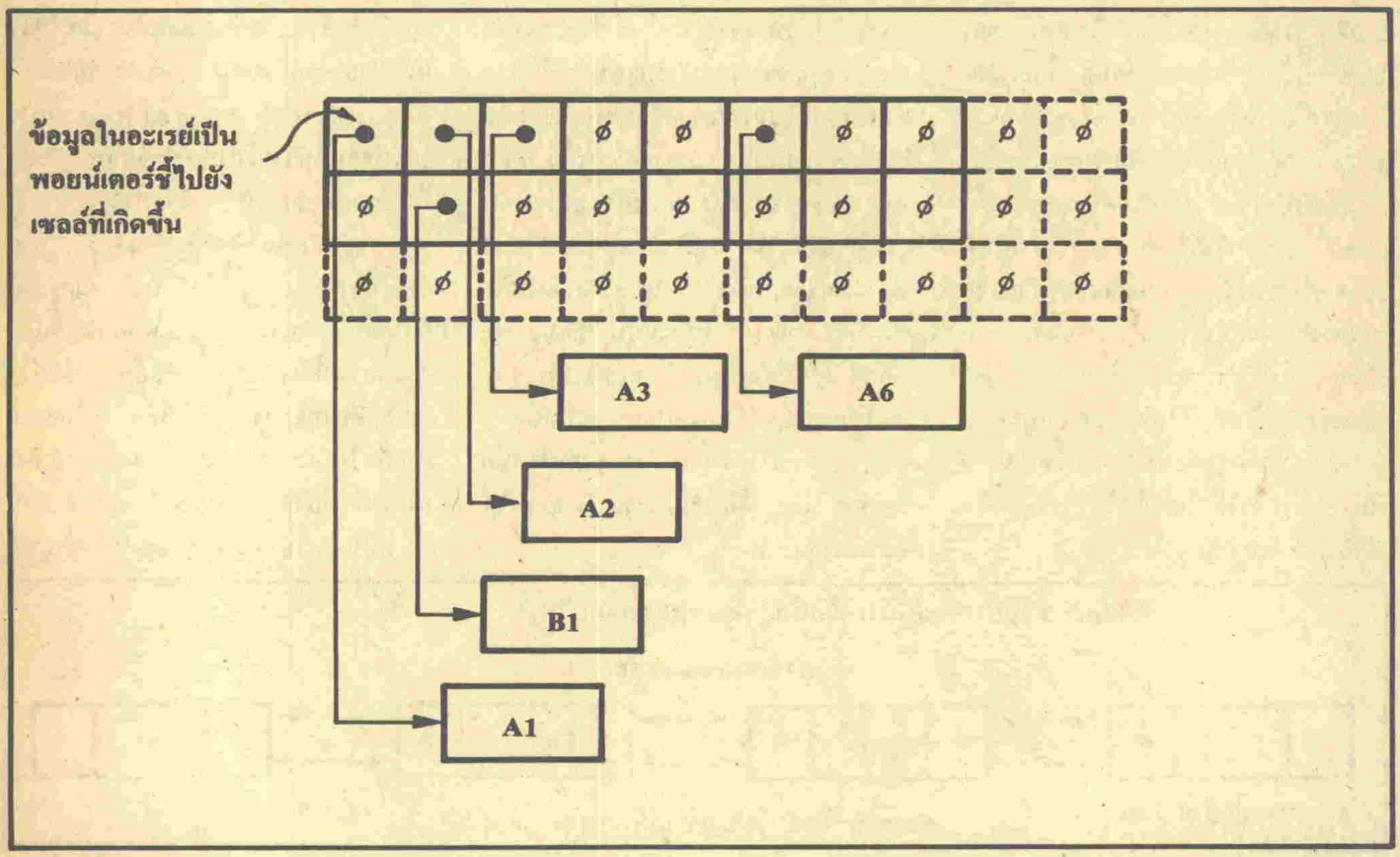

**รูปที่ 2 โครงสร้างข้อมูลแบบ Binary tree**

## ข้อสรุปโครงสร้างข้อมูลทั้ง 3 แบบ

หลังจากที่ศึกษาถึงโครงสร้างโครงข้อมูลของสเปรดชีตทั้งสามแบบแล้ว ต่อไปจะมาวิเคราะห์และสรุปถึงข้อดีข้อเสียของทั้งสามแบบ โดยแบ่งออกเป็น 2 เรื่อง คือเรื่องการใช้หน่วยความจำและความเร็วในการค้นหาเซลล์

### ● ประสิทธิภาพในการใช้หน่วยความจำ

ในเรื่องของการใช้หน่วยความจำนั้นจะพิจารณาเป็น 2 กรณีคือ กรณีที่มีการใช้เซลล์ของสเปรดชีตแบบกระจัดกระจาย




**รูปที่ 3 โครงสร้างข้อมูลแบบ Pointer-arrays**

และการใช้เซลล์แบบเต็มที่หรือเกือบทั้งสเปรดชีต

ในกรณีการใช้เซลล์ของสเปรดชีตแบบกระจัดกระจายนั้น โครงสร้างข้อมูลในลักษณะของ Linked list และ Binary tree จะใช้หน่วยความจำแบบมีประสิทธิภาพสูงกว่าโครงสร้างข้อมูลแบบ Pointer array เพราะว่าหน่วยความจำที่ถูกใช้จริง ๆ คือ เซลล์ที่เกิดขึ้นหรือถูกใช้งานเท่านั้น เซลล์ที่ไม่เกิดขึ้นก็จะไม่กินเนื้อที่ของหน่วยความจำ ผิดกับโครงสร้างข้อมูลแบบ Pointer array ซึ่งเซลล์ที่ไม่ถูกใช้งานก็จะมี Pointer อยู่ในอะเรย์ตลอดเวลา และขนาดของอะเรย์จะต้องถูกกำหนดให้มีขนาดที่เพียงพอที่จะมีพื้นที่หน่วยความจำเหลือสำหรับตัวข้อมูลของเซลล์เองและสำหรับใช้งานอย่างอื่นด้วย ส่วนระหว่าง Linked list และ Binary tree นั้น โครงสร้างข้อมูลทั้งสองแบบใช้จำนวนหน่วยจำต่อเซลล์เท่า ๆ กัน

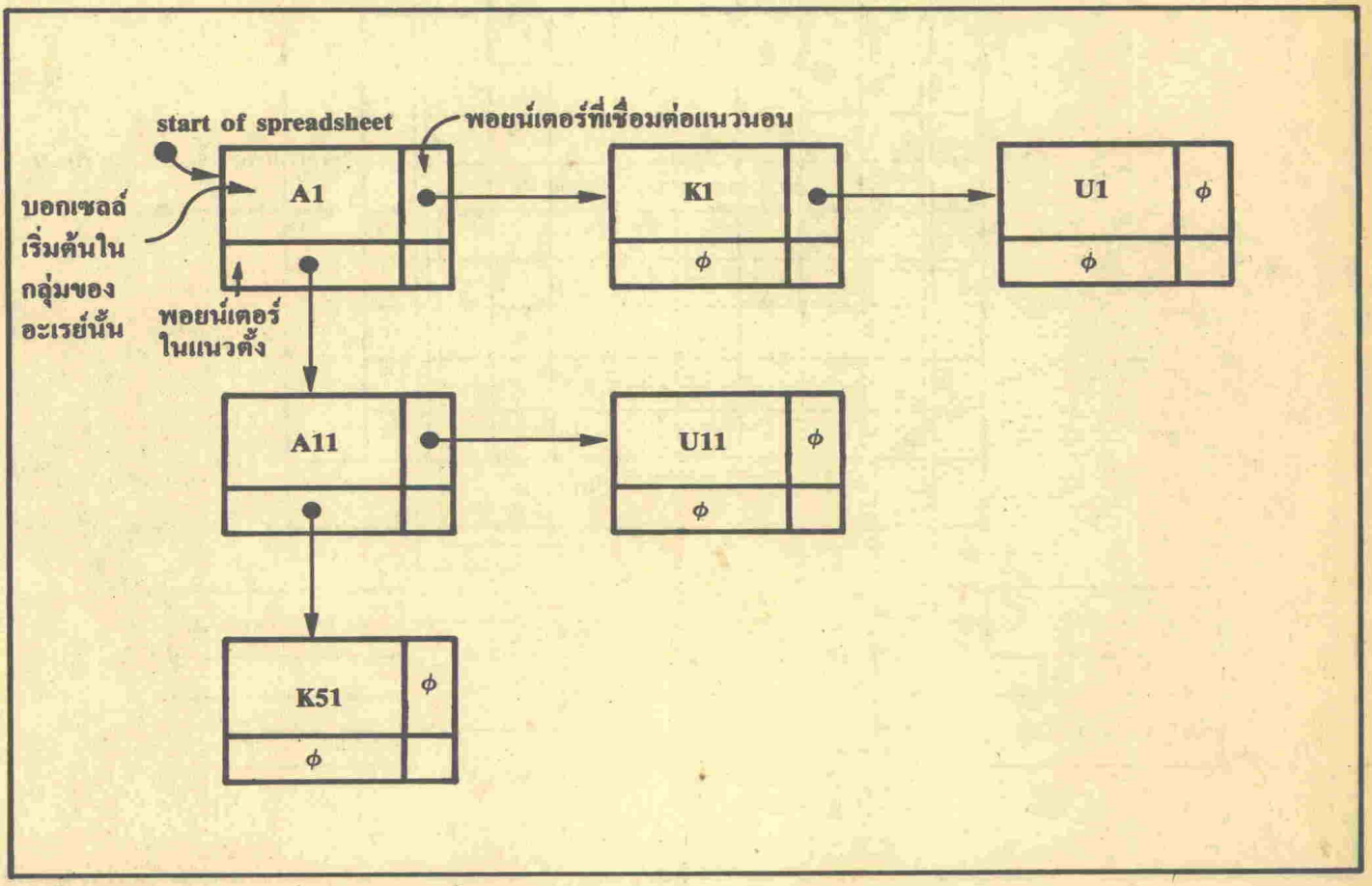
ในกรณีที่มีการใช้งานเซลล์เกือบเต็มสเปรดชีต จะตรงข้ามกับกรณีแรก เพราะว่าโครงสร้างข้อมูลแบบ Pointer-array จะใช้หน่วยความน้อยกว่าโครงสร้างข้อมูลแบบ Linked list และแบบ Binary tree ทั้งนี้เพราะว่าโครงสร้างข้อมูลทั้ง 2 แบบ คือ Linked list และ Binary tree ต้องการพอยน์เตอร์ 2 ตัวต่อข้อมูลหนึ่งเซลล์ในขณะที่ Pointer-array ต้องการพอยน์เตอร์เพียง 1 ตัวต่อเซลล์เท่านั้น เช่นในกรณีใช้เซลล์เต็มที่ 10,000 เซลล์เต็มอะเรย์ ในขณะที่ขนาดของพอยน์เตอร์ขนาด 4 ไบต์ของทั้ง Linked-list และ Binary tree ต้องใช้หน่วยความจำถึง 80,000 ไบต์ แต่ Pointer-array ใช้เพียง 40,000 ไบต์ เท่านั้น ประหยัดกว่าถึง 40,000 ไบต์หรือหนึ่งเท่าตัว

### ● ความเร็วในการค้นหาเซลล์

ในเรื่องของความเร็วในการค้นหาเซลล์นั้น แบบ Pointer-array จะค้นหาได้เร็วที่สุดและง่ายกว่าแบบ Linked-list และ Binary tree เพราะในการเข้าถึงพอยน์เตอร์เซลล์เพียงแต่บอกตัวชี้ (index) แถวและคอลัมน์เท่านั้นเอง

## รวบรวมข้อดีของโครงสร้างข้อมูลทั้งหมดเป็นโครงสร้างข้อมูลแบบใหม่

เมื่อรวบรวมข้อดีของแต่ละโครงสร้างที่กล่าวมาแล้ว ในเรื่องของการประหยัดหน่วยความจำและเรื่องของความเร็วในการค้นหาเซลล์ จะได้เป็นโครงสร้างข้อมูลอันใหม่ซึ่งมีลักษณะเป็น Linked list ของกลุ่มอะเรย์ของพอยน์เตอร์ (pointer array) ขนาดของอะเรย์เลือกตามความเหมาะสม (ในตอนนี้จะใช้ 10×10) การเรียงต่อของ Linked list ก็จะเป็นดังในรูปที่ 4




รูปที่ 4 โครงสร้างข้อมูลของสเปรดชีตที่รวบรวมข้อดีมาจากโครงสร้างของ

การเรียงต่อของกลุ่มอะเรย์ของพอยน์เตอร์ (Pointer array) นั้น ในแนวนอนกลุ่มอะเรย์ของพอยน์เตอร์ที่ต่อกันจะมีชื่อแถวเท่ากันและจะเรียงลำดับกันตามชื่อของคอลัมน์ เช่นในรูปมีลำดับเป็น A1→K1→U1 ส่วนในแนวตั้ง จะมีการเชื่อมกันเฉพาะหัวแถวของแนวนอนเท่านั้น ซึ่งการต่อก็จะเรียงลำดับต่อกันตามชื่อแถว ส่วนโครงสร้างของตัวกลุ่มอะเรย์ (Pointer array) จะต้องมีส่วนพิเศษที่เพิ่มขึ้นมาคือส่วนบอกถึงเซลล์เริ่มต้นอะเรย์นั้น ดังรูปที่ 5

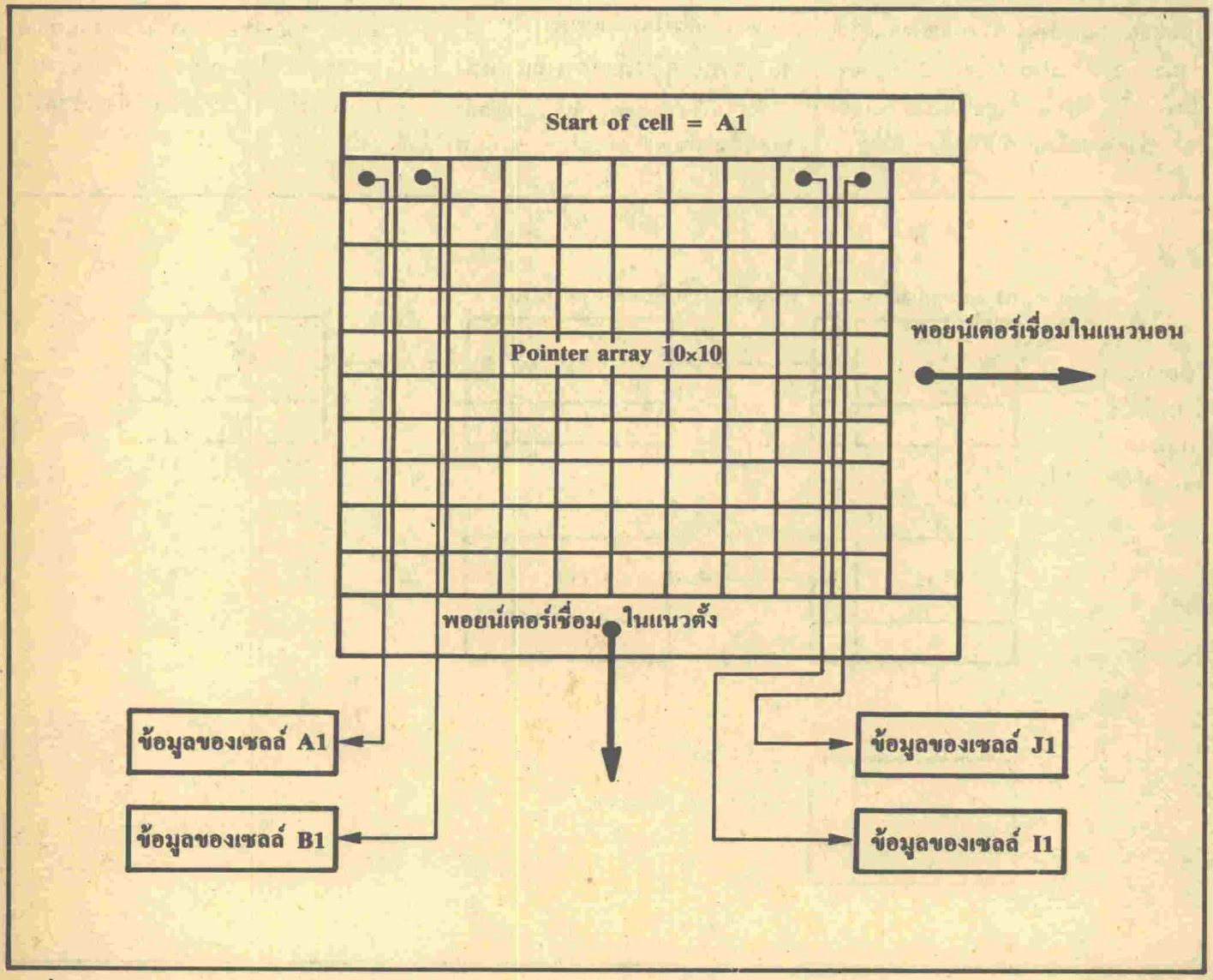
จากลักษณะโครงสร้างข้อมูลสเปรดชีตอันใหม่นี้ ในกรณีที่มีการใช้งานเซลล์ของสเปรดชีตเพียงหนึ่งเซลล์ หน่วยความจำจะถูกใช้งานมีขนาดเท่ากับอะเรย์ของพอยน์เตอร์ 1 อะเรย์ทันที (ดังในรูปที่ 5) เพราะเราจะถือว่าการใช้งานสเปรดชีตโดยทั่วไปแล้วมักจะใช้งานเป็นกลุ่มเซลล์ที่อยู่ติด ๆ กัน ซึ่งขนาดของอะเรย์จะเลือกเท่าไรอะไรก็ได้ตามความเหมาะสม โครงสร้างข้อมูลนี้จึงใช้หน่วยความจำอย่างค่อนข้างมีประสิทธิภาพ

ส่วนเรื่องของความเร็วในการค้นหาเซลล์นั้น ถ้าเรามองตามรูปที่ 4 การค้นหาทำได้ง่าย โดยจะเริ่มค้นหาตามชื่อของแถวก่อน เมื่อพบแถวจึงค้นหาตามคอลัมน์ต่อ การค้นหานั้นเราสามารถจะข้ามกลุ่มเซลล์ได้ถึงทีละ 100 เซลล์ (ในกรณีอะเรย์มีขนาด 10×10) ซึ่งทำให้ค้นหาได้รวดเร็วพอสมควร

## การคำนวณในสเปรดชีต (recalculation)

ในสเปรดชีตนั้น เมื่อใส่สูตรการคำนวณลงไปในเซลล์ การคำนวณของสเปรดชีตทำอย่างไรบ้าง การใช้สูตรอ้างอิงเซลล์แบบวนรอบ (cyclic) นั้น จะเกิดอะไรขึ้น เป็นสิ่งที่เราจะศึกษาถึงแนวความคิดในการคำนวณของสเปรดชีตเป็นเรื่องต่อไป

ก่อนอื่นมาทำความรู้จักกับคำศัพท์ในการคำนวณบางคำก่อน การคำนวณ




รูปที่ 5 โครงสร้างของอะเรย์ของพอยน์เตอร์

(recalculate) ในโลตัส 1-2-3 จะมีด้วยกัน 3 แบบ คือ

1. การคำนวณแบบเรียงลำดับ ของสูตรที่อ้างอิงถึงกันตามธรรมชาติคือ การคำนวณจะเริ่มต้นคำนวณในเซลล์ที่มีการเปลี่ยนแปลงแล้ว จึงคำนวณเซลล์อื่นที่มีผลขึ้นตรงกับเซลล์ที่มีค่าเปลี่ยน ตามความสัมพันธ์ของสูตรที่อ้างอิงกัน ซึ่งในบทความนี้จะยกตัวอย่างอัลกอริทึม ให้ศึกษากัน 2 วิธี

2. การคำนวณแบบเรียงลำดับตามคอลัมน์ (Columnwise) การคำนวณจะเริ่มตั้งแต่เซลล์ A1 ในแต่ละคอลัมน์จนหมดโดยใช้คอลัมน์เป็นหลัก

3. การคำนวณแบบเรียงลำดับแถว (Rowwise) ก็จะคล้ายกับแบบที่ 2 แต่ใช้แถวเป็นหลัก

ส่วนคำว่า Iteration หมายถึงจำนวนในการคำนวณวนรอบเมื่อใส่สูตรอ้างอิงแบบวนรอบ (circular references)

ในไมโครซอฟต์เอกเซลนั้น จะมีการคำนวณเพียงแบบเดียวคือการคำนวณแบบเรียงลำดับตามธรรมชาติเท่านั้น แต่สามารถจะกำหนดค่า Iteration หรือจำนวนครั้งในการวนรอบ และกำหนดค่าที่ต้องให้เปลี่ยนแปลงสูงสุดของเซลล์ในแต่ละครั้งของการคำนวณวนรอบ ซึ่งในโลตัส 1-2-3 จะกำหนดได้เพียงค่า Iteration เท่านั้น

ในการใช้สูตรอ้างอิงแบบวนรอบนั้น ตัวโปรแกรมสเปรดชีตจะตรวจสอบพร้อมกับเตือนผู้ใช้ว่าขณะนี้ได้มีการใส่สูตรที่จะทำให้เกิดการคำนวณแบบเป็นวนรอบ และจะไม่ยอมคำนวณให้ (ในเอกเซลจะไม่คำนวณ แต่ในโลตัส 1-2-3 จะคำนวณให้เพียงครั้งเดียว) จนกว่าจะมีการกำหนดให้มีการคำนวณแบบวนรอบได้พร้อมกับตั้งค่า Iteration เสียก่อน ในเอกเซลนั้นสามารถจะตั้งค่าการเปลี่ยนแปลงสูงสุดของการคำนวณแต่ละรอบในทุก ๆ เซลล์ได้ด้วย การคำนวณจะสิ้นสุดเมื่อมีการคำนวณจนถึงรอบที่กำหนดใน Iteration หรือการเปลี่ยนในเซลล์บางเซลล์น้อยกว่าค่าการเปลี่ยนแปลงที่กำหนด (ในไมโครซอฟต์เอกเซล) ขอยกตัวอย่างการใช้สูตรแบบอ้างอิงเป็นวนรอบสักตัวอย่างหนึ่งเพื่อความเข้าใจ สมมติว่า เราต้องหาค่าของอนุกรม $\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\ldots+\frac{1}{2^n}$ ว่ามีค่าเป็นเท่าไร ก็สามารถหาได้โดยใส่สูตรลงในสเปรดชีตตามนี้

ที่เซลล์ A1 : *A1 + 1*

A2 : *1/(2^A1)+A2*

และกำหนดค่า Iteration เป็น 50 คำตอบที่ได้คือ 1 ซึ่งการใช้สูตรแต่อ้างอิงวนรอบนั้นใช้ในการแก้ปัญหาทางคณิต-ศาสตร์

คราวนี้เรามาศึกษาถึงวิธีการคำนวณของสเปรดชีตดูกัน ซึ่งจะมีแนวความคิดให้ศึกษากัน 2 วิธีคือ วิธีการหาลำดับการคำนวณโดยวิธี Topological Sort และวิธีการคำนวณตามความสัมพันธ์ระหว่างเซลล์ในสูตรที่ใส่

## ● Topological Sort

รายละเอียดของอัลกอริทึมนี้จะไม่กล่าวถึงในบทความนี้ ผู้อ่านที่สนใจหาอ่านได้จากหนังสือ Advanced Programiag : Design and Structure Using Pascal อยู่ในบทที่ 14 เรื่อง Minicalc - A Miniature Spreadsheet จากสูตรความสัมพันธ์ระหว่างเซลล์ที่มีการอ้างอิงถึงกันจะถูกมองเป็นลักษณะ directional graph ดังตัวอย่างรูปที่ 6

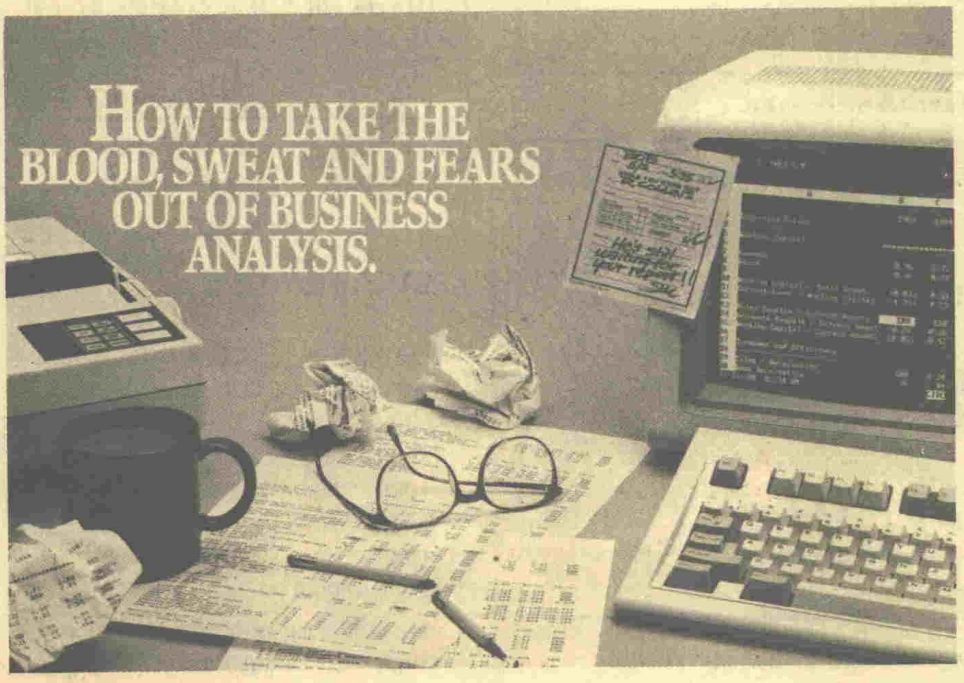
จาก directional graph ของสูตรในเซลล์ตัวอย่างรูปที่ 6 เราจะพบว่าการคำนวณค่าในเซลล์ A1 นั้น มีเส้นทางการคำนวณได้หลายทาง เช่น เริ่มคำนวณจาก

**A2→B1→C2→B5→A3→A1**

เส้นทางดังกล่าวข้างต้น ค่าที่คำนวณได้ใน A1 จะผิด เพราะค่าใน B5 นั้นมีค่าขึ้นอยู่กับ A3 แต่ A3 ยังไม่ได้ถูกคำนวณเลยทั้ง ๆ ที่ค่าใน A3 ขึ้นอยู่กับ A2 และ B1 เส้นทางการคำนวณที่ถูกต้องจริง ๆ คือ

**A2→B1→C2→A3→B5→A1**

เส้นทางที่ถูกต้องนี้หาได้จากวิธี Topological Sort ในเรื่อง directional graph ซึ่งถ้าหากนำวิธีนี้มาใช้ในสเปรดชีต โครงสร้างข้อมูลของเซลล์อาจจะต้องเพิ่มโครงสร้างพิเศษที่จะใช้ในอัลกอริทึมนี้นอกจากโครงสร้างสำหรับเก็บค่าตัวเลข, ตัวอักษร



และสูตร อัลกอริทึมนี้สามารถที่จะตรวจสอบได้ด้วยว่าสูตรที่อ้างอิงถึงกันในสเปรดชีตนั้น มีการอ้างอิงแบบวนรอบอยู่หรือไม่

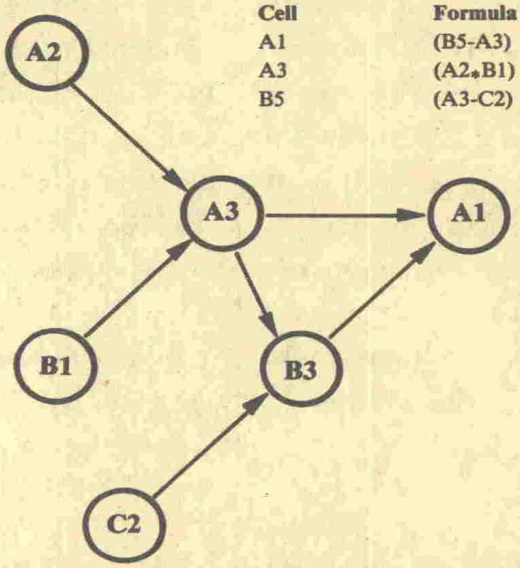
## ● การคำนวณโดยอาศัยความสัมพันธ์ของสูตร

การคำนวณโดยวิธีนี้นั้น คือเก็บความสัมพันธ์ระหว่างเซลล์ที่ถูกอ้างอิงโดยสูตรเอาไว้ในตัวเซลล์ว่า เซลล์แต่ละเซลล์นั้นมีค่าขึ้นอยู่กับเซลล์ใดบ้าง โดยที่โครงสร้างของตัวเซลล์จะมีโครงสร้างส่วนพิเศษสำหรับเก็บชื่อเซลล์ที่ตัวมันเองมีผลกระทบต่อเซลล์นั้น โครงสร้างส่วนพิเศษนี้จะมีลักษณะเป็น Linked list ซึ่งจะขอเรียกส่วนนี้ว่า Dependent List ตัวอย่างเช่น ที่เซลล์ A12 มีสูตรเป็น A10–A11 ดังนั้นที่เซลล์ A10 และ A11 ในส่วนของโครงสร้างพิเศษ ซึ่งเป็น Linked list จะเก็บชื่อเซลล์ A12 เอาไว้ดังรูปที่ 7

ในกรณีที่เซลล์ A10 หรือ A11 มีการเปลี่ยนแปลงค่า การคำนวณจะเริ่มโดยดึงชื่อเซลล์ที่อยู่ใน Dependent list มาเป็นเซลล์ที่ต้องเริ่มคำนวณ เหมือนเริ่มคำนวณเซลล์ที่มีผลลัพธ์ขึ้นตรงต่อเซลล์ที่มีการเปลี่ยนค่านั้นเอง อัลกอริทึมการคำนวณแสดงในรูปที่ 8

การคำนวณในลักษณะนี้สามารถจะทำการคำนวณสูตรที่มีการอ้างอิงเซลล์แบบวนรอบได้ด้วย โดยเพิ่มโครงสร้างเข้าไปในตัวเซลล์อีกโครงสร้างหนึ่ง เพื่อเป็นส่วนบอกหรือเป็นตัวนับรอบในการคำนวณผ่านเซลล์นั้นอีกครั้ง

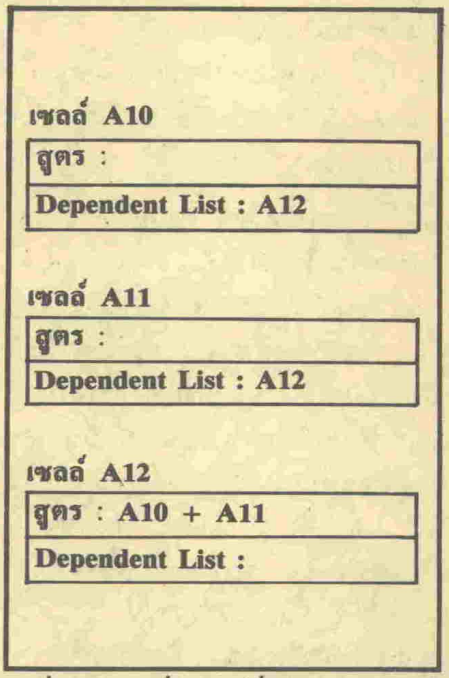
เนื้อเรื่องในบทความนี้อาจจะเป็นเรื่องที่เข้าใจยากสักหน่อย แต่ถ้าผู้อ่านท่านใดสนใจจะศึกษาต่อก็ค้นหาหนังสือในส่วนเอกสารอ้างอิงมาอ่านเพิ่มเติมได้ หวังอย่างยิ่งว่าผู้อ่านคงได้ความรู้เกี่ยวกับการทำงานของโปรแกรมสเปรดชีตเพิ่มขึ้นจากหนังสืออ้างอิงที่ให้ไว้




**รูปที่ 6 Directional Graph ที่ได้จากสูตรของเซลล์ในตัวอย่าง**

```
Procedure recalc. Takes os input row and column numbers of a cell
        to recolcuate
Begin
    If the cell contains a valid formula, then
        Evaluate the formula;
        If the formula results in a value or status that differ from
            from what the cell currently contains, then
            Update the cell's value and status;
            Display the cell;
            For each row and column number on the cell's
                dependentcolls list.
                Call recalc with that row and column
End
```

**รูปที่ 8 แสดงอัลกอริทึมการคำนวณซ้ำโดยอาศัยความสัมพันธ์ระหว่างเซลล์**




**รูปที่ 7 แสดงชื่อเซลล์ที่เก็บในโครงสร้างพิเศษ เมื่อใส่สูตรที่เซลล์ A12 เป็น A10 + A11**

## เอกสารอ้างอิง


1. HERBERT SCHILDT, 1986. **Advanced C**, Osborne McGraw-Hill
2. JONATHAN AMSTERDAM, 1986. *"Build A Spreadsheet Programs"*, **Byte**, Vol.13 No.7 (JULY 1986)
3. LAWRENCE H.Miller, **Advanced Programing : Design and Structure Using Pascal**